week-4

Aim: To build a simple feed forward Neural network to recognize hand written character using MNIST dataset

objective:-

1, To understand the architecture of a feed forward Neural Network.

2, to apply supervised learning for image classification.

3, To preprocess the models accuracy and analysis the performance.

procedure:-

1, import libraries (tensor flow, keras, matplotlib

2, load MNIST Dataset

3, Normalize the data (pixel values)

4, one_hot encode labels (0-9 into length of 10

5, define model architecture:-
 → Flattern layer
 → Dense hidden layer
 → Dense output layer

6, compile model with optimizer, loss function metrics

7, train the model on training data.

8, evalute performance on test data

9, plot accuracy graph for train and validation set)

pseudo code:-

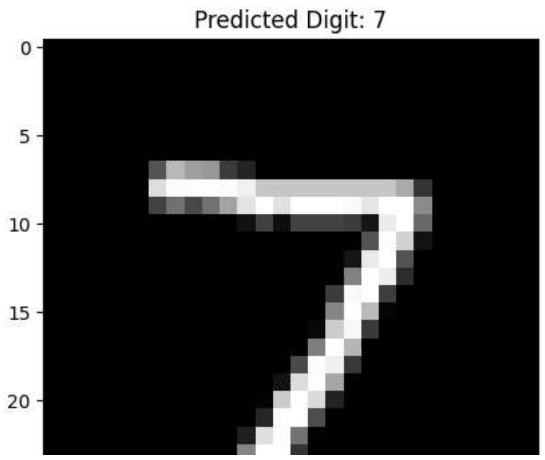
```
start
Import tensor flow, keras, matplotlib.
load MNIST dataset
Normalize input images to range (0,1)
```

convert labels to one hot encoding
Initialize sequential model
Add flatten layer for 28x28 input.
Add dense hidden layer with Relu activation
Add dense output layer with softmax activation
entropy' loss, accuracy metrics..
train model for 5 epochs.
Evalute model on test data.
plot training vs validation accuracy
predict labels for first 5 test samples
display predicted lables with images
END

~~output~~
observations:-
* the model achieved ~ 97-98% accuracy on the MNIST test dataset with just 5 epochs of training
* accuracy improved steadily with each epoch, indicating effective learning.
* Prediction on unseen data matched the actual digits in most cases.
* errors occured mainly on digits that were poorly written.

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize images (0-255 → 0-1)
x_train = x_train.reshape((60000, 784)).astype('float32') / 255.0
x_test  = x_test.reshape((10000, 784)).astype('float32') / 255.0

# One-hot encode labels (0-9)
y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test, 10)

model = models.Sequential()
model.add(layers.Dense(128, activation='relu', input_shape=(784,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    x_train, y_train,
    epochs=10,
    batch_size=128,
    validation_split=0.2,
    verbose=2
)

test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print("\n✅ Test Accuracy:", round(test_acc*100, 2), "%")

import numpy as np

predictions = model.predict(x_test)
sample = 0  # Change index to test different samples
plt.imshow(x_test[sample].reshape(28,28), cmap='gray')
plt.title("Predicted Digit: {}".format(np.argmax(predictions[sample])))
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` ar
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
375/375 - 3s - 8ms/step - accuracy: 0.8910 - loss: 0.3838 - val_accuracy: 0.9397 - val_loss: 0.2025
Epoch 2/10
375/375 - 2s - 4ms/step - accuracy: 0.9545 - loss: 0.1579 - val_accuracy: 0.9609 - val_loss: 0.1391
Epoch 3/10
375/375 - 3s - 7ms/step - accuracy: 0.9676 - loss: 0.1099 - val_accuracy: 0.9646 - val_loss: 0.1202
Epoch 4/10
375/375 - 2s - 4ms/step - accuracy: 0.9759 - loss: 0.0822 - val_accuracy: 0.9664 - val_loss: 0.1175
Epoch 5/10
375/375 - 3s - 7ms/step - accuracy: 0.9810 - loss: 0.0639 - val_accuracy: 0.9712 - val_loss: 0.0964
Epoch 6/10
375/375 - 2s - 4ms/step - accuracy: 0.9853 - loss: 0.0507 - val_accuracy: 0.9716 - val_loss: 0.0953
Epoch 7/10
375/375 - 2s - 4ms/step - accuracy: 0.9875 - loss: 0.0411 - val_accuracy: 0.9744 - val_loss: 0.0912
Epoch 8/10
375/375 - 2s - 4ms/step - accuracy: 0.9904 - loss: 0.0329 - val_accuracy: 0.9726 - val_loss: 0.0941
Epoch 9/10
375/375 - 3s - 7ms/step - accuracy: 0.9924 - loss: 0.0269 - val_accuracy: 0.9716 - val_loss: 0.1039
Epoch 10/10
375/375 - 2s - 5ms/step - accuracy: 0.9936 - loss: 0.0222 - val_accuracy: 0.9728 - val_loss: 0.0979

✅ Test Accuracy: 97.44 %
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```


Predicted Digit: 7